[redacted]

Optimally profiling and tracing programs

Full text Pdf (1.27 MB)

Source **Annual Symposium on Principles of Programming Languages** archive
Proceedings of the 19th ACM SIGPLAN-SIGACT symposium on Principles of programming languages table of contents
Albuquerque, New Mexico, United States
Pages: 59 - 70
Year of Publication: 1992
ISBN:0-89791-453-8

Authors Thomas Ball
James R. Larus

Sponsors SIGACT: ACM Special Interest Group on Algorithms and Computation Theory
SIGPLAN: ACM Special Interest Group on Programming Languages

Publisher ACM Press New York, NY, USA

Additional Information: abstract references citings index terms collaborative colleagues peer to peer

Tools and Actions: Discussions Find similar Articles Review this Article
Save this Article to a Binder Display in BibTex Format

DOI Bookmark: Use this link to bookmark this Article: http://doi.acm.org/10.1145/143165.143180
What is a DOI?

↑ ABSTRACT

This paper presents algorithms for inserting monitoring code to profile and trace programs. These algorithms greatly reduce the cost of measuring programs. Profiling counts the number of times each basic block in a program executes and has a variety of applications. Instruction traces are the basis for trace-driven simulation and analysis, and are also used in trace-driven debugging. The profiling algorithm chooses a placement of counters that is optimized—and frequently optimal—with respect to the expected or measured execution frequency of each basic block and branch in the program. The tracing algorithm instruments a program to obtain a subsequence of the basic block trace—whose length is optimized with respect to the program's execution—from which the entire trace can be efficiently regenerated. Both algorithms have been implemented and produce a substantial improvement over previous approaches. The profiling algorithm reduces the number of counters by a factor of two and the number of counter increments by up to a factor of four. The tracing algorithm reduces the file size and overhead of an already highly optimized tracing system by 20-40%.

↑ REFERENCES

Note: OCR errors may be found in this Reference List extracted from the full text article. ACM has opted to expose the complete List rather than only correct and linked references.

1 T. Ball and j. R. Larus, "'Optimally Profiling and Tracing Programs," Technical Report #1031, University of Wisconsin, Madison (July 1991).

2 Robert F. Cmelik , Shing I. Kong , David R. Ditzel , Edmund J. Kelly, An analysis of MIPS and SPARC instruction set utilization on the SPEC benchmarks, ACM SIGARCH Computer Architecture News, v.19 n.2, p.290-302, Apr. 1991

3 Systems Performance Evaluation Cooperative, SPEC Newsletter (K. Mendoza, editor) 1(1)(1989).

4 Joseph A. Fisher , John R. Ellis , John C. Ruttenberg , Alexandru Nicolau, Parallel processing: a smart compiler and a dumb machine, Proceedings of the symposium on Proceedings of the SIGPLAN '84 symposium on compiler construction, p.37-47, June 17-22, 1984, Montreal, Canada

5 S.L. Graham, P. B. Kessler, and M. K. McKusick, "An Execution Profiler for Modular Programs," Software Practice and Experience 13 pp. 671-685 (1983).

6 Jeff L. Kennington , Richard V. Helgason, Algorithms for Network Programming, John Wiley & Sons, Inc., New York, NY, 1980

7 D.E. Knuth and F. R. Stevenson, "Optimal Measurement Points for Program Frequency Counts," BIT 13 pp. 313-322 (1973).

8 J. R. Larus, Abstract execution: a technique for efficiently tracing programs, Software—Practice & Experience, v.20 n.12, p.1241-1258, Dec. 1990

9 S. Maheshwari, "Traversal marker placement problems are NP-complete," Report No. CU-CS-092-76, Dept. of Computer Science, University of Colorado, Boulder, CO (1976).

10 Scott McFarling, Procedure merging with instruction caches, ACM SIGPLAN Notices, v.26 n.6, p.71-79, June 1991

11 B. P. Miller and J. D. Choi, "A Mechanism for Efficient Debugging of Parallel Programs," Proceedings of the ACM SIGPLAN 1988 Conference on Programming Language Design and implementation (SIGPLAN Notices) 23(7)pp. t35-144 (June 1988).

12 W. G. Morris, CCG: a prototype coagulating code generator, ACM SIGPLAN Notices, v.26 n.6, p.45-58, June 1991

13 Karl Pettis , Robert C. Hansen, Profile guided code positioning, ACM SIGPLAN Notices, v.25 n.6, p.16-27, Jun. 1990

14 S. Pottle, private communication. October 1991.

15 R. L. Probert, "Optimal Insertion of Software Probes in Well-Delimited Programs," IEEE Transactions on Software Engineering SE-8(1) pp. 34-42 (January, 1975).

16 C.V. Ramamoorthy, K. H. Kim, and W. T. Chen, "Optimal Placement of Software Monitors Aiding Systematic Testing," IEEE Transactions on Software Engineering SE-I(4)pp. 403-410 (December, 1975).

17 Alan Dain Samples, Profile-driven compilation, University of California at Berkeley, Berkeley, CA, 1992

18 V. Sarkar, Determining average program execution times and their variance, ACM SIGPLAN Notices, v.24 n.7, p.298-312, July 1989

19 Alan Jay Smith, Cache Memories, ACM Computing Surveys (CSUR), v.14 n.3, p.473-530, Sept.

1982

20 MIPS Computer Systems, inc., UMIPS-V Reference Manual (pixie and pixstats), MIPS Computer Systems, Sunnyvale, CA (1990).

21 Robert Endre Tarjan, Data structures and network algorithms, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1983

↑ CITINGS *40*

Jason R. C. Patterson, Accurate static branch prediction by value range propagation, ACM SIGPLAN Notices, v.30 n.6, p.67-78, June 1995

Jörn Schneider , Christian Ferdinand, Pipeline behavior prediction for superscalar processors by abstract interpretation, ACM SIGPLAN Notices, v.34 n.7, p.35-44, July 1999

Benjamin Zorn , Dirk Grunwald, Evaluating models of memory allocation, ACM Transactions on Modeling and Computer Simulation (TOMACS), v.4 n.1, p.107-131, Jan. 1994

Peter S. Magnusson, Efficient instruction cache simulation and execution profiling with a threaded-code interpreter, Proceedings of the 29th conference on Winter simulation, p.1093-1100, December 07-10, 1997, Atlanta, Georgia, United States

Todd A. Proebsting , Charles N. Fischer, Demand-driven register allocation, ACM Transactions on Programming Languages and Systems (TOPLAS), v.18 n.6, p.683-710, Nov. 1996

David A. Barrett , Benjamin G. Zorn, Using lifetime predictors to improve memory allocation performance, ACM SIGPLAN Notices, v.28 n.6, p.187-196, June 1993

Seth J. White , David J. DeWitt, QuickStore: a high performance mapped object store, ACM SIGMOD Record, v.23 n.2, p.395-406, June 1994

Eui-Young Chung , Luca Benini , Giovanni De Micheli, Source code transformation based on software cost analysis, Proceedings of the international symposium on Systems synthesis, September 30-October 03, 2001, Montréal, P.Q., Canada

Seth J. White , David J. DeWitt, QuickStore: a high performance mapped object store, The VLDB Journal — The International Journal on Very Large Data Bases, v.4 n.4, October 1995

David A. Sykes , Brain A. Malloy, The design of an efficient simulator for the Pentium Pro processor, Proceedings of the 28th conference on Winter simulation, p.840-847, December 08-11, 1996, Coronado, California, United States

Youfeng Wu , James R. Larus, Static branch frequency and program profile analysis, Proceedings of the 27th annual international symposium on Microarchitecture, p.1-11, November 30-December 02, 1994, San Jose, California, United States

Dirk Grunwald , Benjamin Zorn , Robert Henderson, Improving the cache locality of memory allocation, ACM SIGPLAN Notices, v.28 n.6, p.177-186, June 1993

Robert H. B. Netzer , Mark H. Weaver, Optimal tracing and incremental reexecution for debugging long-running programs, Proceedings of the ACM SIGPLAN '94 conference on Programming language design and implementation, p.313-325, June 20-24, 1994, Orlando, Florida, United States

Thomas Ball , James R. Larus, Optimally profiling and tracing programs, ACM Transactions on Programming Languages and Systems (TOPLAS), v.16 n.4, p.1319-1360, July 1994

Todd A. Proebsting , Christopher W. Fraser, Detecting pipeline structural hazards quickly, Proceedings of the 21st ACM SIGPLAN-SIGACT symposium on Principles of programming languages, p.280-286, January 16-19, 1994, Portland, Oregon, United States

Steven M. Kurlander , Charles N. Fischer, Minimum cost interprocedural register allocation, Proceedings of the 23rd ACM SIGPLAN-SIGACT symposium on Principles of programming languages, p.230-241, January 21-24, 1996, St. Petersburg Beach, Florida, United States

Steven P. Reiss, Software visualization in the desert environment, ACM SIGPLAN Notices, v.33 n.7, p.59-66, July 1998

Thomas Ball , James R. Larus, Branch prediction for free, ACM SIGPLAN Notices, v.28 n.6, p.300-313, June 1993

Norman Ramsey, Relocating machine instructions by currying, ACM SIGPLAN Notices, v.31 n.5, p.226-236, May 1996

Todd A. Proebsting , Charles N. Fischer, Probabilistic register allocation, ACM SIGPLAN Notices, v.27 n.7, p.300-310, July 1992

Mark E. Crovella , Thomas J. LeBlanc, Parallel performance prediction using lost cycles analysis, Proceedings of the 1994 ACM/IEEE conference on Supercomputing, November 14-18, 1994, Washington, D.C.

J. K. Hollingsworth , B. P. Miller, Parallel program performance metrics: a comprison and validation, Proceedings of the 1992 ACM/IEEE conference on Supercomputing, p.4-13, November 16-20, 1992, Minneapolis, Minnesota, United States

Mark E. Crovella , Thomas J. LeBlanc, Parallel performance using lost cycles analysis, Proceedings of the 1994 conference on Supercomputing, p.600-609, December 1994, Washington, D.C., United States

Brad Calder , Dirk Grunwald, Reducing indirect function call overhead in C++ programs, Proceedings of the 21st ACM SIGPLAN-SIGACT symposium on Principles of programming languages, p.397-408, January 16-19, 1994, Portland, Oregon, United States

Mary F. Fernández, Simple and effective link-time optimization of Modula-3 programs, ACM SIGPLAN Notices, v.30 n.6, p.103-115, June 1995

Koen De Bosschere , Saumya Debray , David Gudeman , Sampath Kannan, Call forwarding: a simple interprocedural optimization technique for dynamically typed languages, Proceedings of the 21st ACM SIGPLAN-SIGACT symposium on Principles of programming languages, p.409-420, January 16-19, 1994, Portland, Oregon, United States

Marc L. Corliss , E. Christopher Lewis , Amir Roth, DISE: a programmable macro engine for customizing applications, ACM SIGARCH Computer Architecture News, v.31 n.2, May 2003

G. Ramalingam, Data flow frequency analysis, ACM SIGPLAN Notices, v.31 n.5, p.267-277, May 1996

Kemal Ebcioglu , Randy D. Groves , Ki-Chang Kim , Gabriel M. Silberman , Isaac Ziv, VLIW compilation techniques in a superscalar environment, Proceedings of the ACM SIGPLAN '94 conference on Programming language design and implementation, p.36-48, June 20-24, 1994, Orlando, Florida, United States

Hiralal Agrawal, Dominators, super blocks, and program coverage, Proceedings of the 21st ACM SIGPLAN-SIGACT symposium on Principles of programming languages, p.25-34, January 16-19, 1994, Portland, Oregon, United States

Dinesh C. Suresh , Walid A. Najjar , Frank Vahid , Jason R. Villarreal , Greg Stitt, Profiling tools for hardware/software partitioning of embedded applications, ACM SIGPLAN Notices, v.38 n.7, July 2003

Steven K. Reinhardt , Mark D. Hill , James R. Larus , Alvin R. Lebeck , James C. Lewis , David A. Wood, The Wisconsin Wind Tunnel: virtual prototyping of parallel computers, ACM SIGMETRICS Performance Evaluation Review, v.21 n.1, p.48-60, June 1993

Karim Harzallah , Kenneth C. Sevcik, Predicting application behavior in large scale shared-memory multiprocessors, Proceedings of the 1995 ACM/IEEE conference on Supercomputing (CDROM), p.53-es, December 04-08, 1995, San Diego, California, United States

Hira Agrawal, Efficient coverage testing using global dominator graphs, ACM SIGSOFT Software Engineering Notes, v.24 n.5, p.11-20, Sept. 1999

Eric A. Brewer , William E. Weihl, Developing parallel applications using high-performance simulation, ACM SIGPLAN Notices, v.28 n.12, p.158-168, Dec. 1993

John Whaley, Partial method compilation using dynamic profile information, ACM SIGPLAN Notices, v.36 n.11, p.166-179, 11/01/2001

Amer Diwan , David Tarditi , Eliot Moss, Memory subsystem performance of programs using copying garbage collection, Proceedings of the 21st ACM SIGPLAN-SIGACT symposium on Principles of programming languages, p.1-14, January 16-19, 1994, Portland, Oregon, United States

Amer Diwan , David Tarditi , Eliot Moss, Memory system performance of programs with intensive heap allocation, ACM Transactions on Computer Systems (TOCS), v.13 n.3, p.244-273, Aug. 1995

Robert Wahbe , Steven Lucco , Thomas E. Anderson , Susan L. Graham, Efficient software-based fault isolation, ACM SIGOPS Operating Systems Review, v.27 n.5, p.203-216, Dec. 1993

Matthew F. Parkinson , Sri Parameswaran, Profiling in the ASP codesign environment, Proceedings of the eighth international symposium on System synthesis, p.128-133, September 13-15, 1995, Cannes, France

↑ INDEX TERMS

Primary Classification:

D. Software

↪ **D.2** SOFTWARE ENGINEERING

↪ **D.2.5** Testing and Debugging

↪ **Subjects:** Tracing

Additional Classification:

F. Theory of Computation

↪ **F.2** ANALYSIS OF ALGORITHMS AND PROBLEM COMPLEXITY

↪ **F.2.2** Nonnumerical Algorithms and Problems

↪ **Subjects:** Computations on discrete structures

G. Mathematics of Computing

↪ **G.2** DISCRETE MATHEMATICS

↪ **G.2.2** Graph Theory

↪ **Subjects:** Trees

General Terms:
Algorithms, Performance

↑ **Collaborative Colleagues:**

Thomas Ball:
Stephen Adams
Glenn Ammons
David Atkins
David L. Atkins
A. Michael Berman
Hans Boehm
Glenn Bruns
Sagar Chaki
Satish Chandra
Yih-Farn Chen
Yih-farn Chen
Trishul M. Chilimbi
Kenneth Cox
Manuvir Das
F. Douglis
Fred Douglis
Stephen G. Eick
Todd Graves
Todd L. Graves
Susan Horwitz
Dean F. Jerding
Eleftherios Koutsofios
Krishna Kunchithapadam
James Larus
James R. Larus
Sorin Lerner
Rupak Majumdar
Peter Mataga
Todd Millstein
Audris Mockus
Mayur Naik
Andreas Podelski
Sriram K. Rajamani
Thomas Reps
Mooly Sagiv
Mark Seigle
Michael Siff
John T. Stasko
Frank Tip
Westley Weimer

James R. Larus:
Alexander Aiken
Glenn Ammons
Thomas Ball
Rastislav Bodík
Rastislav Bodík
Satish Chandra
Trishul M. Chilimbi
Trishul Madhukar Chilimbi
Bob Davidson
Stephen G. Eick
Hersham El-Rewini
Babak Falsafi
Jack Grimes
Paul N. Hilfinger
Mark D. Hill
Lorenz Huelsbergen
Steven Huss-Lederman
Phil Laplante
Alvin R. Lebeck
James C. Lewis
Mike Litzkow
David Mandelin
Bertrand Meyer
Barton P. Miller
Shubhendu S. Mukherjee
David Notkin
Subbarao Palacharla
Michael Parkes
Gustav Pomberger
Mike Potel
Dave Power
Wolfgang Pree
Steven K. Reinhardt
Brad Richards
Bradley Richards
Bradley Eric Richards
Anne Rogers
Joel Saltz
Richard D. Schlichting
Eric Schnarr
Eric C. Schnarr
Ioannis Schoinas
Shamik D. Sharma
Madhusudhan Talluri
Ron Vetter
Guhan Viswanathan
Bruce W. Weide
David A. Wood
Youfeng Wu
Zhichen Xu

↑ **Peer to Peer - Readers of this Article have also read:**

- Data structures for quadtree approximation and compression
  Communications of the ACM 28, 9
  Hanan Samet

- A hierarchical single-key-lock access control using the Chinese remainder theorem
  Proceedings of the 1992 ACM/SIGAPP Symposium on Applied computing
  Kim S. Lee , Huizhu Lu , D. D. Fisher

- 3D representations for software visualization
  Proceedings of the 2003 ACM symposium on Software visualization
  Andrian Marcus , Louis Feng , Jonathan I. Maletic

- Probabilistic surfaces: point based primitives to show surface uncertainty
  Proceedings of the conference on Visualization '02
  Gevorg Grigoryan , Penny Rheingans

- Efficient simplification of point-sampled surfaces
  Proceedings of the conference on Visualization '02
  Mark Pauly , Markus Gross , Leif P. Kobbelt

ferencing of memory references to expose patterns in program execution that are not immediately available in the raw trace. Using a compaction scheme that looks for and encodes long common sub-sequences of bytes results in compressed trace files less than 10% of original size for full traces, and on the order of 0.5% the size of the original trace for instruction-only traces.

Acknowledgements: The final form of this paper been improved considerably by constructive input from several people. Thanks to Paul Hilfinger, David Wood, Charlie Farnum, the referees, and especially Alan Smith for their comments. Any problems that remain are mine, and are probably the result of not paying closer attention to their suggestions.

8 Bibliography

1. AGARWAL, A. Trace Compaction Using Cache Filtering with Blocking. draft, 1987.
2. HAMMERSTROM, D. W. AND DAVIDSON, E. S. Information Content of CPU Memory Referencing Behavior. *Proceedings 4th Annual Symposium on Computer Architecture* (March 1977).
3. KNUTH, D. *The TeXbook*. Addison Wesley, Reading, MA, 1984.
4. MANDELBROT, B. B. *The fractal geometry of nature*. W. H. Freeman and Company, San Francisco, CA, 1983.
5. PUZAK, T. R. Analysis of Cache Replacement-Algorithms. University of Massachusetts, PhD Dissertation, February 1985.
6. SAMPLES, A. D. Code Reorganization for Instruction Caches. Computer Science Division, EECS, University of California, Berkeley, Technical Report UCB/CSD 88/447, 1988.
7. SMITH, A. J. Two Methods for the Efficient Analysis of Memory Address Trace Data. *IEEE Transactions on Software Engineering*, *SE-3*, 1 (January 1977).
8. WALL, D. Register Windows vs. Register Allocation. *SIGPLAN '88 Conference on Programming Language Design and Implementation*, *23*, 7 (June 22-24, 1988), 67–78.
9. WELCH, T. A. A Technique for High Performance Data Compression. *IEEE Computer*, *17*, 6 (June 1984), 8–19.
10. ZIV, J. AND LEMPEL, A. A Universal Algorithm for Sequential Data Compression. *IEEE Transactions on Information Theory*, *23* (1976), 75–81.
11. ZIV, J. AND LEMPEL, A. Compression of Individual Sequences via Variable-rate Coding. *IEEE Transactions on Information Theory*, *24* (1978), 530–536.